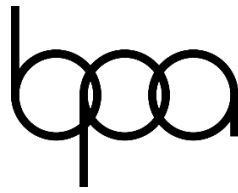


Contestant ID: \_\_\_\_\_

Time: \_\_\_\_\_

Rank: \_\_\_\_\_



**BUSINESS  
PROFESSIONALS  
of AMERICA**  
Giving Purpose to Potential

# **PYTHON PROGRAMMING (355)**

## **REGIONAL 2026**

**APPLICATION KNOWLEDGE:**

**TOTAL POINTS** \_\_\_\_\_ **(250 points)**

**Test Time: 90 minutes**

## GENERAL GUIDELINES

*Failure to follow any of these rules may result in disqualification:*

1. **Submission Requirements:** Contestants must submit this test booklet along with any printouts.
2. **Permitted Items:** Only the equipment, supplies, and materials specified for this event are allowed in the testing area. Previous BPA tests and sample tests (whether handwritten, photocopied, or typed) are not permitted.
3. **Electronic Devices:** Electronic devices will be monitored according to ACT standards.

## Event Requirements:

You will have ninety (90) minutes to complete your work.

Your name and/or school name should *not* appear on work you submit for grading.

## Event Guidelines:

1. Create a folder using your *Contestant ID* as the name of the folder.
2. Create and name your file, *RegionalPython.py*, is the file in which you will write your solution. Your file *must* be a Python file (a .py file).
3. If submitting your test with a flash drive, copy your folder to the flash drive.
4. If submitting your test online, zip your folder and upload the zipped folder to the test submission website.
5. You will need to use a local Python IDE to complete this exam. No online interpreters for Python are allowed.
6. Graders will not modify or debug your code — it must run as-is.
7. Submissions that do *not* contain source code and/or files other than Python files will *not* be graded.

## What you will be doing:

1. Include your Contestant ID as a comment at the top of the main source code file.
2. Completing a menu of various tasks, demonstrating a knowledge of essential Python skills.

## Development Standards:

Your code will be evaluated on structure, readability, and maintainability:

- Use consistent variable naming conventions (e.g., snake\_case).
- Use functions to organize major parts of your program.
- Each function must include comments explaining its purpose, parameters, and return values.
- Use meaningful input validation and error handling.
- Demonstrate clean and logical program structure with appropriate indentation and readability.
- Close all files properly and include necessary comments.

### **Overview of Challenge**

- You are developing a Disc Golf Tournament Manager program that allows a user to manage scores for a 9-hole round.
- Your solution should collect data, perform calculations, analyze performance, and export results to a file.
- Your goal is to demonstrate both programming logic and practical Python skills.

### **Disc Golf Tournament Manager Requirements**

#### Requirement 1: Player Setup

- Ask how many players are participating.
- Prompt the user to enter each player's name and store them in a list.
- Display a confirmation message summarizing the players.
- Input validation: must be at least one player, and names cannot be blank.

#### Requirement 2: Score Entry

- For each player, request a score for each of the 9 holes.
- Store the results in a nested structure (e.g., dictionary or list of lists).
- Validate that each entry is a valid integer between 1 and 10.
- Re-prompt on invalid input.
- Confirm completion after all players' scores have been entered.

#### Requirement 3: Score Calculations

- Calculate each player's total and average score.
- Display a formatted leaderboard showing player name, total score, and average score.
- Example:

Player Name	Total Score	Average
-----	-----	-----
Jordan	34	3.78
Taylor	39	4.33

- Use appropriate data formatting and rounding.

#### Requirement 4: Tournament Analysis

- Determine the winner (lowest total score).
- Compute the overall average score across all players.
- Identify the hardest hole — the one with the highest average score among all players.
- Display all analysis results clearly in both console output and the results file.
- Use functions for each major analysis component.

Requirement 5: Data Storage

- Save all results in a properly formatted file named *tournament\_results.txt*.
- Include:

Disc Golf Tournament Results

-----

Players: 4

Winner: Jordan (Total: 34)

Leaderboard:

Jordan: 34 (Avg: 3.78)

Taylor: 39 (Avg: 4.33)

...

Hardest Hole: Hole 5 (Average: 4.8)

Overall Average Score: 4.12

- Include the date and time of tournament generation using the datetime module.
- Ensure all file handling (open/write/close) is correct.

Requirement 6: Program Design & Structure

- Modularize your program using multiple functions (e.g., `input_scores()`, `calculate_totals()`, `analyze_tournament()`, `save_results()`).
- Avoid repeated code through loops or functions.
- Demonstrate logical flow and maintainable structure.

<b>Scoring Rubric</b>			
<b>Category</b>	<b>Description</b>	<b>Points</b>	<b>Score</b>
Project Setup & Submission	Folder, file naming, correct file types	10	
Program Execution	Runs without errors and meets task requirements	25	
Player Setup	Prompts for number of players, collects names accurately	15	
Score Entry	Validates and records all hole scores for each player	30	
Score Calculations	Computes totals and averages correctly	40	
Tournament Analysis	Determines winner, hardest hole, overall averages	50	
Data Storage	Outputs correctly formatted tournament report file	30	
Program Design & Structure	Modular code, reusable functions, readability	35	
Comments & Documentation	Clear, useful comments and docstrings throughout	10	
Naming Conventions & Readability	Consistent, descriptive naming and code clarity	5	
File Handling	Proper file open/write/close procedures	10	
<b>TOTAL POSSIBLE POINTS</b>		<b>250</b>	